6) The KD encodes the random number and its Bluetooth address with its secret key (3080) and adds the ticket. The result is sent to the LD (3090, 4110).

7) The LD decrypts the encrypted number and address using the receiver public key in the ticket (4120).

8) If the decryption succeeds (meaning the key was correct) (4130), the decrypted address matches the KD's address (4140), the decrypted, number matches the previously sent random number (4150), the ticket is successfully verified (5000, 4160), the access rights of the user allow him to get in now (4170), and no TID or public key in the ticket is blacklisted (or, in the case of n-use tickets, above their use limit)(4180), then the user is authorized to open the lock(4190, 3100, 3110, 3120, 3140, 4200). If the ticket is n-use, and unshared, the KD may increase its use count to keep track of how many uses are left (3115). Also, the LD may include in the Access Granted message (4190) the number of times the ticket has been used. If the ticket is shared, this allows the KD to update its use counter to reflect the current situation.

9) Closing the lock again could be based on a simple timeout (lock stays open for a predefined time interval), but preferably the LD would use radio signal strength to get some measure of the distance to the KD, and when signal strength is sufficiently weak (4210), close the lock again (4220).

10) If the ticket's access limits specify that it is a n-use ticket (4230), then all TIDs in the ticket with a limited number of uses (nested tickets are included) are stored on the blacklist with a count of 1. If they have a validity limit date, that is stored on the blacklist as well, to allow it later to be purged of obsolete information. If some were already blacklisted, then the count for those is increased in the blacklist (4240).

Figure 5 shows an embodiment of the verification process of a ticket.

1) If the ticket is not derivative (5020), the LD decrypts the checksum (5040) with the public key that corresponds to granter KID (5030), and verifies that the decryption succeeded (meaning that key was correct) (5050), the checksum is correct (5060, 5070), and that the granter KID is authorized to grant the ticket (access rights are stored within the LD) (5080, 5090, 5100).

2) If the ticket is derivative (5020), the LD decrypts the checksum with the public key of the granter (5120), which is the receiver public key in the original (nested) ticket (5110), and verifies that the decryption succeeded (meaning that key was correct) (5130), and the checksum is correct (5140,5150). It then recursively verifies the original ticket (5000, 5160), and checks that it allowed at least one level of derivative tickets (5170). It also checks that the level of derivative tickets allowed increases by at least one at each recursion level (5180), that limits on derivative tickets are observed (5190), that derivation limits are included in the derivation limits of the nested ticket(5200), and that access limits are always included in the access limits of the nested ticket(5210). The last two inclusion checks are necessary to prevent people from creating derivative tickets that are less restrictive that the ticket they possess.

3) If the ticket passes all the checks, it has been successfully verified (5220).

Keys can be added, removed, and their access rights can be modified. These are simple database operations. Keys are structured as a forest (a group of tree hierarchies). Each key must have been authorized either by an LD PIN (root keys), or by another key. The parent of a key is the key that authorized it.

Keys can be removed or their access rights modified only by keys above them in the hierarchy. Possibly a KD can also remove itself from a lock. PIN authorization allows everything ("root access"). A key cannot have wider access rights than its parent.

If a key is removed because it has been compromised, all keys below it in the hierarchy should also be removed.

For security reasons, users may want to change their secret keys periodically. Update of LDs can be done automatically, by allowing public keys in LDs to be updated by the owner of the keys. The KD must then store both the old and the new RSA key pair until all LDs have been updated.

The KD can remove the old RSA key pair once all LDs have been updated. This requires that the KD stores the LIDs of all LDs it has access to. As this increases the memory requirements of the KD, it would be an optional feature. The other option is that the old RSA key pair is removed manually by the user.

It is often not desirable for a door to unlock because someone with a KD walks nearby on the inside. The lock should be shielded against radio signals from that direction (doors can be opened from the inside without unlocking).

However, if the door should block movement also from the inside, the radio signals should be restricted so that there is a very small area immediately next to the door where KDs will open the door.

Also, to prevent unauthorized people from sneaking in when someone with a KD (that allows access) walks past the door, either the locations where KDs are effective on the outside should be limited to the immediate vicinity of the door, or confirmation should be required. Still another possibility is that an LD only opens the door if the KD stays close (determined by signal strength) for some time.

Some advantages for this system are: